AI COURSEWORK 2&3
03/04/2024

THA22530230
SUDIP THAPA MAGAR

Sustainable Transportation Solutions

Introduction:

This document explores the Python program that I developed for my AI Coursework 2, aimed at devising sustainable transportation solutions. The assignment's core objective was to engineer an AI-powered navigation system for a self-driving ride-share car fleet, enhancing route efficiency and reducing carbon emissions across the UK.

Defining the Map and the Searching Problem:

The program begins with defining a graph structure to represent the network of cities and the distances between them as given in the assignment brief. This graph is stored as a dictionary, where each key is a city, and its value is another dictionary mapping adjacent cities to their distances. This nested dictionary approach was chosen for its efficiency in representing and accessing the graph's edges and nodes. The Get_User_Input function allows users to dynamically select their start and end points, using a while loop to ensure valid input, showcasing careful error handling.

Defining the Searching Algorithms:

To explore the graph, four distinct algorithms were implemented: DFS, BFS, A*, and Dijkstra's Algorithm. DFS and BFS are implemented using recursive calls and dequeues, respectively, illustrating different approaches to graph traversal. The A* algorithm uses a priority queue with a heuristic function, demonstrating a more sophisticated search strategy that considers both the cost so far and an estimate to the goal. I chose Dijkstra's Algorithm due to its effectiveness in finding the shortest paths in weighted graphs that don't have negative edges. Its approach of using a priority queue to carefully select the next node makes it particularly well-suited for optimizing routes in our transportation network's weighted graph.

Showing the Results:

Accumulate_Search_Results function illustrates a commitment to sustainability, calculating and displaying carbon emissions, air quality, and green space percentages. These metrics were carefully selected to provide users with a comprehensive view of their travel choices' ecological implications, empowering them to make informed decisions.

Conclusion:

Creating this program went beyond just an assignment; it was an exploration of how AI can be used for sustainable purposes. I had to dive into both the theory behind search algorithms and how to apply them in real life. This project has strengthened my conviction in technology's ability to tackle urgent environmental issues, motivating me to keep working on projects that blend AI with sustainability efforts.

Cancer Treatment Decision Simulator

Overview:

This document is intended to provide a detailed overview of the "Cancer Treatment Decision Simulator," a Python-based application that utilizes a Tic Tac Toe model to simulate treatment decisions in a healthcare context. The purpose of this simulation is to highlight the importance of strategic decision-making in cancer treatment and its impact on patient health.

Modules and Libraries:

The simulation uses Python's `random` library for generating random numbers and the `prettytable` library for formatting and displaying the scoreboard in a tabular format.

Functions

Print_Board(board_positions, show_note = True):

This function prints the current state of the Tic Tac Toe board, representing different aspects of a patient's health status. It optionally displays a note on the significance of 'X' and 'O' symbols in the context of treatment decisions and cancer progression.

Print_Scoreboard(score_board):

Utilizes the `PrettyTable` class from the `prettytable` library to display the current scores of the clinician and the AI (representing cancer's progression) in a well-structured table format.

Check_Victory(player_positions, current_player):

Determines if the current player (clinician or AI) has won the game by checking against a predefined set of winning combinations.

Check_For_Draw(player_positions):

Checks if the game has reached a draw by verifying if all positions on the board are filled without any player winning.

Check_Game(current_board):

Evaluates the current state of the board to identify if there's a winning condition for either player, used primarily by the AI to make decisions.

Mini_Max(current_board, depth, is_maximizing):

Implements the MiniMax algorithm to simulate an optimal move for the AI based on the current board state, maximizing the AI's chance of winning while minimizing the clinician's chances.

Find_Best_Move(current_board):

Determines the best possible move for the AI on the current board by evaluating each possible move using the MiniMax algorithm.

AI_Move(game_board, player_positions):

Executes the best move for the AI by updating the board and the AI's positions based on the output from `Find_Best_Move`.

Game_Intro():

Prints an introductory message explaining the simulation's premise, the roles of the clinician ('X') and the AI ('O'), and the ethical considerations in treatment decisions.

Initialize_Game(current_player):

Initiates and manages the game loop, alternating turns between the clinician and the AI, and checking for victory or draw conditions after each move.

Main:

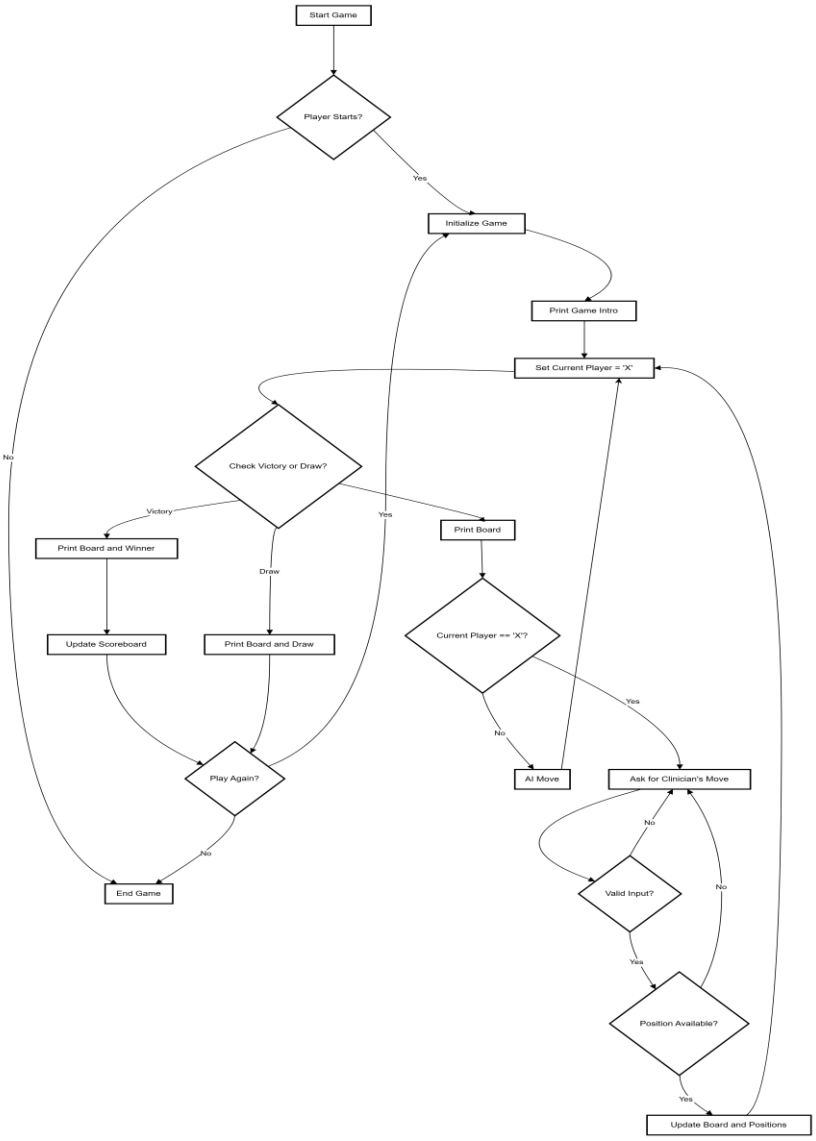
The entry point of the application, which sets up the game environment, including the introduction, initializing player names and scores, and managing the game start and continuation prompts.

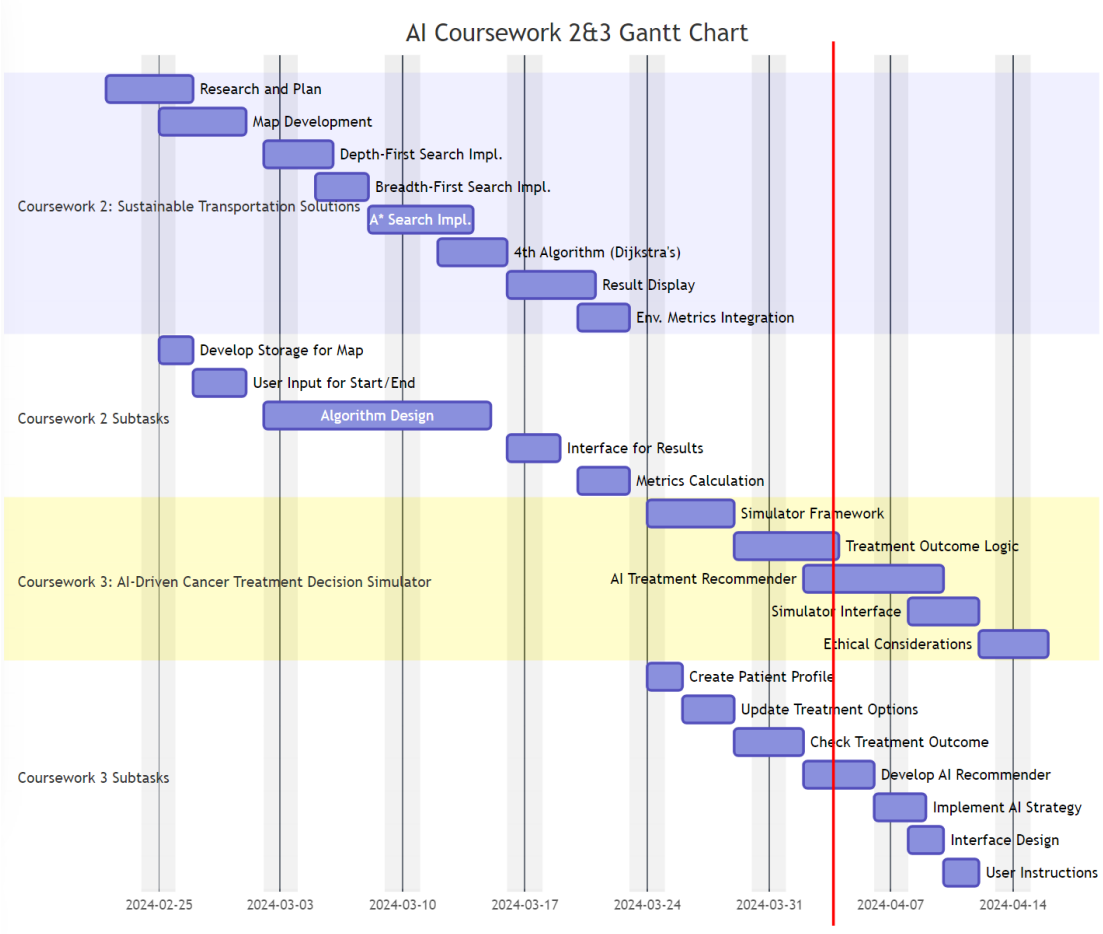
Gameplay Instructions:

Upon starting the simulation, you will be greeted with an introduction to the game's context and objectives.
2. The game board will be displayed, and you, playing as the clinician ('X'), will be prompted to choose a position on the grid (1-9) for your move.
3. After your move, the AI, representing cancer's progression ('O'), will automatically make its move.
4. The game proceeds with alternating turns until either a victory condition is met for any player or the game ends in a draw.
5. After the game concludes, you will be presented with the option to start a new game or exit.

Conclusion:

This "Cancer Treatment Decision Simulator" serves as an educational tool to underscore the complexity and critical nature of decision-making in cancer treatment. Through the lens of a familiar game, it invites healthcare professionals and student doctors to consider the myriad factors that influence patient care outcomes.




AI Coursework 2&3 Gantt Chart
Research and Plan
Map Development
Depth-First Search Impl.
Breadth-First Search Impl.
A* Search Impl.
4th Algorithm (Dijkstra's)
Result Display
Env. Metrics Integration
Coursework 2: Sustainable Transportation Solutions
Develop Storage for Map
User Input for Start/End
Algorithm Design
Interface for Results
Metrics Calculation
Coursework 2 Subtasks
Simulator Framework
Treatment Outcome Logic
AI Treatment Recommender
Simulator Interface
Ethical Considerations
Coursework 3: AI-Driven Cancer Treatment Decision Simulator
Create Patient Profile
Update Treatment Options
Check Treatment Outcome
Develop AI Recommender
Implement AI Strategy
Interface Design
User Instructions
Coursework 3 Subtasks
2024-02-25
2024-03-03
2024-03-10
2024-03-17
2024-03-24
2024-03-31
2024-04-07
2024-04-14